# Open Source Wifi Hotspot Implementation

Tyler Sondag and Jim Feher

*The goal of this paper is to describe a design—including the hardware, software, and configuration—for an open source wireless network. The network designed will require authentication. While care will be taken to keep the authentication exchange secure, the network will otherwise transmit data without encryption.*

Wireless networks are an essential tool for providing service for colleges and libraries. This paper will explain the setup of a wireless network using open-source software and inexpensive commodity hardware. Open-source software was employed exclusively. This allowed for flexibility in design and reduction in expense while also providing a platform for students to learn more about the internal workings of the system by examining particular sections of code in which they have interest. Standard commodity hardware was used as a means of saving cost. This should allow others to repeat this design with a minimum of funding.

The purpose of a network, like any resource, is to provide a service for those who own it; in this case, the patrons of a library, or students, faculty, and staff at a college. To ensure that this network serves its owners, users will be required to authenticate before gaining access. Once authenticated, the central captive portal can provide different levels of service for specific user groups, including guest access, if desired. For this system, ease of access for users was the primary concern; other than using the Secure Socket Layer for authentication, the remainder of the traffic was unencrypted.

Other than the base nodes, the remaining access points were connected to each other using a wireless connection in order to avoid physically connecting all access points across campus and to further reduce the expense for the deployment of the network. This was accomplished using the WDS (wireless distributed system) feature on the wireless routers. All access points connect to a centralized set of servers that provide: DHCP, Web-caching proxy, DNS caching, radius, Web server, a captive portal, and logging of network traffic.

## Hardware

Requirements for the network were relatively modest, using inexpensive wireless routers along with several Linux servers built upon older Pentium 3 desktop systems. Linksys WRT54GS routers were chosen as the access points as they are inexpensive, readily available, and possess the ability to run custom open-source firmware. Other access points could be used; however, the configuration suggestions are specific to the WRT54GS and may not apply to other hardware. The routing functions of the WRT54GS were not used in this implementation. The servers need not be anything special; older hardware will work just fine. For this implementation, decommissioned 900 MHz units with 512MB of RAM and 40GB hard drives were used.

## Wireless router software

In order to provide the functionality required, the units had their firmware flashed with an open-source, Linux-based operating system available from OpenWrt for the Linksys routers (http://www.openwrt.org). Support is also available for other wireless devices. "The firmware from OpenWrt provides a fully writable file system with package management. This allows developers the freedom to customize the devices by choosing only the packages and software that are necessary for their applications."[1] As the routers have limited storage, being able to hand select only the necessary components is a definite advantage.

## Server software

For the operating system on the servers, Fedora Core was chosen.[2] Fedora provides the Yellow Dog Updater, Modified (yum), which eases the updating of all packages installed on the system, including kernel updates.[3] This aids security by providing a platform for easily and frequently updating the system. Fedora Core is an open-source distribution that is available for free. Fedora Core also comes with many other open-source packages that were used in this design, such as the Apache Web server. While the designers had more familiarity with Fedora, other distributions are also available that provide similar benefits (Suse, Ubuntu, OpenBSD, Debian, etc.). The server was run in command line mode with no graphical user interface in order to reduce the load on the server and save space on the hard drive.

## Captive portal

In order to require authentication before gaining access to the network, a captive portal was used. Some of the

**Jim Feher** (jdfeher@mckendree.edu) is an Associate Professor of Computer Science at McKendree College in Lebanon, Illinois. **Tyler Sondag** (tnsondag@mckendree.edu), is a senior in Computer Science at McKendree College.

desired features in the choice of the captive portal were: encrypted authentication, traffic logging, and the ability to provide different levels of service for different user groups. Logging traffic allows the system administrators to identify accounts that have been misusing the network. Those who inadvertently misuse the system or perhaps have had their accounts compromised can have their access temporarily disabled until they can be contacted with instructions concerning acceptable use of the network. As the network must be shared by all, those who habitually abuse the resource can have their accounts permanently disabled. The captive portal should also redirect Web traffic to a login page that is served on the Secure Socket Layer until the user logs in. Chillispot was chosen as it possesses all of the features mentioned above.[4]

## Server layout

As can be seen in appendix A, three servers were used for this implementation. The first server was used as the main router to the Internet. The second server ran a Squid Web caching server.[5] It also ran a DNS caching server and the FreeRADIUS server.[6] The third was used for the captive portal. Three servers were used for various reasons. First, this distributed the load. Second, portions of the network that were not behind the captive portal could more easily use the services on the second server running Squid, DNS, and FreeRADIUS. It should be noted that three independent servers are not required; many of the services could be consolidated on two or even one single server to reduce the hardware requirements. The implementation depends upon the specific needs for the network.

## Server installation

Installing the operating system (Fedora Core) on each server is a relatively straightforward procedure. Each machine was partitioned with 1024 MBs of swap space with the rest of the drive being an ext3 partition with the mount point "/". Only the minimal set of packages required were installed at this time. The first server, server #1 (router), was given three network interfaces, one for the Internet connection, one to connect to a switch that then connects to server #2 (Web/DNS caching and radius) as well as other machines that do not connect through the captive portal, and one connecting to server #3 (captive portal machine). The second server, server #2, only needs one interface, but the third, server #3, requires two interfaces, one for the master wireless access point, and one to connect to the switch connecting this machine to the rest of the network (appendix A). SSH login for root was also disabled at this time for added security.

## Server #1 configuration

For server #1, very little setup was required. Since this server works mainly as a router, the only major items that went into its configuration were the iptables rules, which are shown and described in appendix B.[7] Rules were set up to:

- set up network address translation;
- allow traffic to flow within the network;
- log the traffic from the wireless portion of the network;
- allow for the transparent setup of the Web proxy server; and
- set up port knocking before allowing users to log into the router via SSH.[8]

A reference to this script was placed in the /etc/rc.d/rc.local file so that it would run when the server boots.

Last was the setup of the three network interfaces in the machine. This can be done during system installation or afterwards on the Fedora Core based server by editing the configuration files in the /etc/sysconfig/networking-scripts/ directory. One of the configuration files used in this implementation can be seen in appendix C. Of course the configuration will change as the topology of the network changes.

## Server #2 configuration

The second server required significantly more setup to configure all of the necessary services that it runs. The first service added for this implementation was the Web-caching proxy server, Squid. Squid's default configuration file (/etc/squid.conf) is quite large; fortunately it requires little modification to get a simple server up and running.[9] The changes made for this implementation can be seen in appendix D. The most important lines in this configuration are the last few, which enable it to act as a transparent proxy server, making it invisible to the users and requiring no setup of their browsers.

As there was no need for an authoritative DNS server, just DNS caching for the network, dnsmasq, which is easy to configure and can handle both DHCP services as well as DNS caching, was chosen.[10] In this instance, the captive portal was used to provide DHCP services for the wireless clients; however dnsmasq was used for dynamic clients on the remaining portion of the network. Dnsmasq

is relatively easy to configure, requiring only one change in its default configuration file, which points to the file in which the DNS server addresses are stored, in this case /etc/dnsmasq_resolv.conf.

Next is the configuration of FreeRADIUS server. There are two files that need to be modified for the radius server; both are in the /etc/raddb/ directory. The first is clients.conf (appendix E). In this file at least two clients must be listed, one for localhost (this machine) and one for the captive portal machine. For each machine, a password must be specified as well as the hostname for that machine. This establishes the shared key that is used to encrypt communication between the captive portal and the radius server. The second is the users file (appendix F). In this file, each user for the captive portal system must be listed with his/her password. This implementation also included a class, a session timeout (dhcp lease time), idle timeout, accounting interim interval, and the maximum upload and download speeds. If guest access is required, one or several guest accounts should be added to this file along with entries for the registered users. An entry was added for each access point so that they can obtain an IP address from the DHCP server. Finally for this machine, the interface configuration file was changed according to the network specifications. For this machine the configuration is simple since it only has one interface, and the only requirement for its address is that it be on the same network as the interface on the main router server to which it is connected.

## Server #3 configuration

The third server required the installation of the captive portal software, in this case Chillispot. In order to install Chillispot, if Fedora was used for the base system, it may be possible to install it as a prepackaged binary in the form an RPM package manager (rpm) file. Otherwise, if you find that you need to compile Chillispot from source code, you may need to deviate from a minimal installation of the operating system and base components and also include the GNU compiler collection (gcc).

When installing from source code, first download the code from the Chillispot Web site. Once the code is downloaded, unzipped and untarred, installing the Chillispot daemon is done by entering the directory containing the source files and entering the standard commands:

```
./configure
make
make install
```

When Chillispot is on the system, either by compiling from source or through an rpm file, two more files must be configured and copied to the proper directory, the main configuration file and the login file.

The configuration file, chilli.conf, is located in the directory that contains the source files. Move this file to the /etc/ directory and make the necessary changes. In this implementation, the file required several changes (appendix G). One of the more significant alterations was to change the default network range of 192.168.182.0/24, which would be limited to less than 256 addresses. The address range was for the DHCP server was also expanded to allow for more users. The lower portion of the network range was left to make room for addresses that could be assigned to the wireless access points. An entry was added to allow the access points to obtain a static IP address in that lower range.

After this, settings must be changed for the DNS addresses given out to clients, and the address of the radius server. There is also a setting in the Chillispot configuration file that allows users to access a certain list of domains without logging in. For this implementation, the decision was to allow the users access to the campus network, as well as to the DNS server. Next, the "radiussecret" must be set. This is the same password that was entered into the clients.conf file on the radius server for this machine. It is also necessary to set the address of the page to which users will be directed. Two lines must also be added to allow authentication using the physical or media access control (mac) address for the access points. All of the access points shared a common password. Chillispot passes the physical address of the access point to the radius server along with this password. A separate entry must exist in the radius configuration file for each IP/physical address combination.

For this setup, the redirect page was placed on this server, therefore Apache (using yum) was also installed, and this server's address was added as the Web address for the redirect page (also note that the https module may be required for apache if it does not automatically install). Rather than write a new page at this time, the sample page (hotspotlogin.cgi) from the Chillispot source folder was copied and modified slightly (appendix H). In addition, a Secure Socket Layer (SSL) certificate was installed on this server. This is not necessary, but it helps to avoid the warnings that pop up when a client attempts to access the login page with a browser.

A few iptables rules need to be added. The first command needs to be executed in order to utilize Network Address Translation (NAT) and have the server forward packets to the outside network.

```
/sbin/iptables -t nat -A POSTROUTING -o eth0 \
-j MASQUERADE
```

The next is used to drop all outbound traffic originating from the access points. This prevents anyone spoofing the physical address of the access point from accessing

the Internet, while still allowing the access points and the Chillispot server to communicate for configuration and monitoring.

```
/sbin/iptables -A FORWARD -s 192.168.182.0/24 \
-j DROP
```

These commands need to be executed when the Chillispot machine boots, so they were placed into the /etc/rc.d/rc.local file. It may also be necessary to ensure that the machine can forward network traffic. This can be accomplished with the following command, which is also found as the first executable command from the script in appendix B:

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

Finally, the configuration files for the interfaces were set up.

## OpenWrt installation and configuration

Several ways exist to replace the default Linksys firmware with the OpenWrt firmware.[11] The tftp protocol can be used with both Windows and Linux, and one such method can be found in Appendix I.[12] In addition, other methods for using the standard Web interface can be found on the OpenWrt Web site.[13] There are several versions of the OpenWrt firmware available; the newest version that uses the squashfs filesystem was chosen because it utilizes compression that frees more space on the access point.

OpenWrt comes with a default Web interface that can be used for configuration, however, ssh was enabled and a script using the nvram command was used to configure each access point (see appendix J). Before ssh can be used, you must telnet into the router and change the default password (which for Linksys routers is '*admin*').

NOTE: Even if you decide to use the Web interface, you should still change the default password.

As several services that were installed with the default configuration were not used in the implementation, they were disabled once the firmware was flashed by removing the modules that boot at startup: the Web interface, dnsmasq, and the firewall. This is done by deleting their entries in the /etc/init.d directory. Changes were needed to set the mode of the access point, to turn on and configure the clients needing to use WDS, to set the network information for the access point and then to save these settings. All of the wireless access points that communicate with each other via a wireless connection must have their physical addresses entered using a nvram command. For example, the command used for the main access point for the library would be:

```
nvram set w10_wds="MAC_4_lib1 MAC_4_lib2"
```

All of this is detailed in appendix J. A final set of commands, which were needed for the WRT54GS, are included to allow the access point to obtain its IP address from the DHCP server. These commands may not be necessary depending upon the type of access point used. Since extra wireless access points are available, if an access point fails or is having problems for some reason, it is simply a matter of running a script similar to the one found in the appendix on one of the extra routers and swapping it out.

## Security

Unfortunately this system is not very secure. Only the login credentials are encrypted via SSL. General data packets are in no way encrypted, so any information being transmitted is available to anyone sniffing the channel. WEP and WPA could be used for encryption, but they have known vulnerabilities. Other methods exist for securing the network such as WPA with RADIUS or the use of a Virtual Private Network, however the client setup for such systems may not be considered trivial for the typical user. Therefore it was decided that it was better to inform the users that the data was not being encrypted and let them act accordingly, rather than use encryption with known flaws or invest the time required to train the general population on how to configure their mobile units to use a more secure form of encryption. As the main goal of this particular network was connectivity and not security, it was felt that this was a fair trade-off. As new standards for wireless communication are developed and commodity hardware that supports them becomes available, this may change so that encrypted channels can be employed more easily.

## Conclusion

This implementation is in no way completed. It is a work in progress, with many goals still in mind. Also, as new features are desired, parts of the system will change to accommodate these requirements. Current plans for the future are first to develop scripts to check the status of the access points and display this information to a Web page. These scripts will also notify network administrators when access points go offline. This will help the administrators in making sure the system is up at all times. After this, scripts will be developed to parse the log files to find abusive activity (spamming, viruses, etc). However, the current project as described is complete and has already functioned successfully for nearly a year providing connectivity for the library and portions of the McKendree College campus.

## References and Notes


**1.** OpenWrt, Wireless Freedom. www.openwrt.org (accessed June 16, 2006).

**2.** The Fedora Project. www.fedora.redhat.com (accessed Nov. 29, 2005).

**3.** Yum: Yellow dog Updater, Modified. www.linux.duke.edu/projects/yum (accessed July 22 2006).

**4.** ChilliSpot—Open Source Wireless LAN Access Point Controller. www.chillispot.org (accessed June 23, 2006).

**5.** Squid Web Proxy Cache. www.squid-cache.org (accessed June 1, 2006).

**6.** FreeRADIUS—Building the Perfect RADIUS Server. www.freeradius.org (accessed June 28, 2006).

**7.** Netfilter/iptables Project Homepage—The netfilter.org Project. www.netfilter.org (accessed Aug. 8, 2006).

**8.** Thomas Eastep, "Port Knocking and Other Uses of 'Recent Match.'" www.shorewall.net/PortKnocking.html (accessed Aug. 11, 2006).

**9.** Squid Web Proxy Cache, "SQUID Frequently Asked Questions: Interception Caching/Proxying." www.squid-cache.org/Doc/FAQ/FAQ-17.html (accessed Aug. 8, 2006).

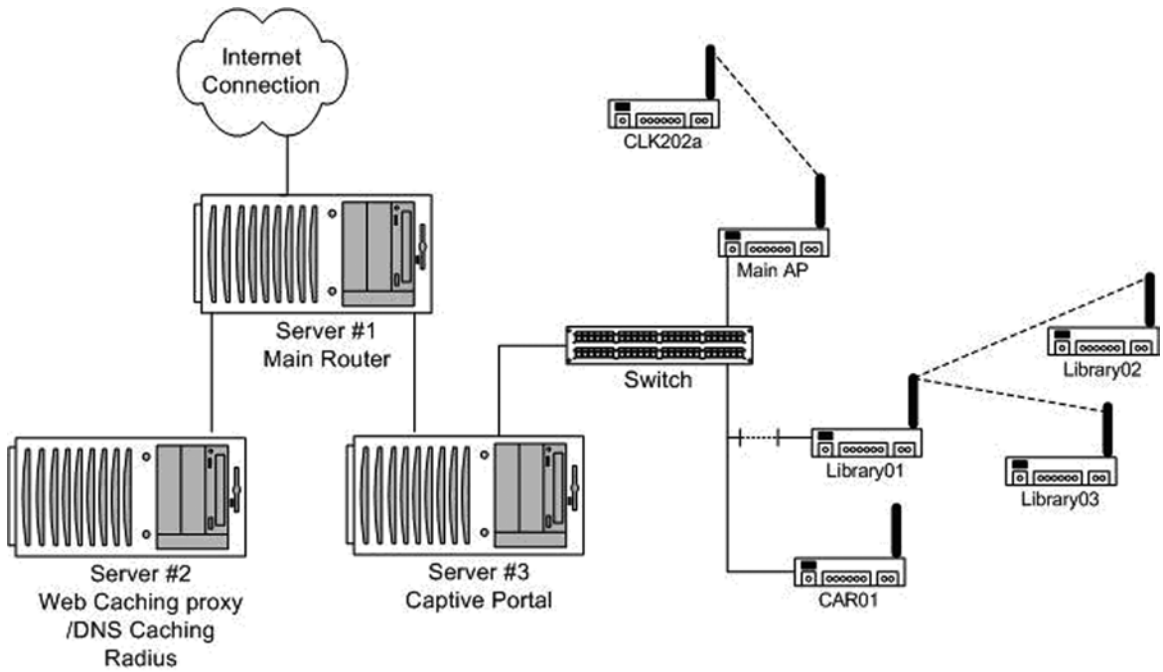
**10.** Dnsmasq—A DNS Forwarder for NAT Firewalls. www.thekelleys.org.uk/dnsmasq/doc.html (accessed June 1, 2006).

**11.** Linksys.com. www.linksys.com (accessed Dec. 15, 2005).

**12.** OpenWrtDocs/Installing/TFTP—OpenWrt. wiki.openwrt.org/OpenWrtDocs/Installing/TFTP?action=show&redirect=OpenWrtViaTfp (accessed Aug. 2, 2006).

**13.** OpenWrtDocs/Installing—OpenWrt. wiki.openwrt.org/OpenWrtDocs/Installing (accessed Aug. 2, 2006).


## Appendix A. Network configuration



## Appendix B. iptables script—Server #1

```
# this particular bit must be set to one to allow the
# network to forward packets
echo "1" > /proc/sys/net/ipv4/ip_forward

# set up path to the internal network from Internet if the
# internal network initiated the connection
iptables -A FORWARD -i eth0 -o eth1 -d 10.4.0.0 \
  -m state --state ESTABLISHED,RELATED -j ACCEPT
# Same for the Chillispot subnet
iptables -A FORWARD -i eth0 -o eth2 -d 10.5.0.0 \
  -m state --state ESTABLISHED,RELATED -j ACCEPT

# allow the internal subnets to communicate with one another
iptables -A FORWARD -i eth1 -d 10.5.0.0 -o eth2 \
  -j ACCEPT
iptables -A FORWARD -i eth2 -d 10.4.0.0 -o eth1 \
  -j ACCEPT

# allow subnet containing server 2 to reach the Internet
iptables -A FORWARD -i eth1 -o eth0 -j ACCEPT

# Chillispot – accept and forward packets
iptables -A FORWARD -i eth2 -s 10.5.3.30 -j ACCEPT

# Set up transparent proxy for wireless network, but allow
# connections that go through to the campus network
# to bypass proxy
iptables -t nat -A PREROUTING -i eth2 ! \
  -d 66.99.172.0/23  -p tcp --dport 80 -s 10.5.0.0/16 \
  -j DNAT --to-destination 10.4.1.90:3128

# nat
iptables -t nat -A POSTROUTING -o eth0 \
  -j MASQUERADE

# simple port knocking to allow port 22 connection adapted
# from www.shorewall.net/PortKnocking.html[1] another
# excellent document can be found at
# www.debian-administration.org/articles/26814
# once connection started let it continue
iptables -A INPUT -m state --state \
ESTABLISHED,RELATED -j ACCEPT

# if name SSH has been set, then allow connection
iptables -A INPUT -p tcp --dport 22 -m recent \
  --rcheck --name SSH -j ACCEPT

# Surround the port that opens ssh so that a sequential port
# scanners will end up closing it right after opening it.
iptables -A INPUT -p tcp --dport 1233 -m recent \
  —name SSH --remove -j DROP
iptables -A INPUT -p tcp --dport 1234 -m recent \
  --name SSH --set -j DROP
iptables -A INPUT -p tcp --dport 1235 -m recent \
  --name SSH --remove -j DROP

# drop all packets that do not match a rule above by default
iptables -A INPUT -j DROP
```

## Appendix C. Server configuration for first network card (ethernet 0)

```
# /etc/sysconfing/networking-scripts/ifcfg-eth0 -
# Server #1
#
DEVICE=eth0
BOOTPROTO=static
BROADCAST=66.128.109.63
HWADDR=00:11:22:33:44:66
IPADDR=66.128.109.60
NETMASK=255.255.255.248
NETWORK=66.128.109.56
ONBOOT=yes
TYPE=Ethernet
```

## Appendix D. /etc/squid.conf—Server #2

```
#default squid port
http_port 3128

# settings changed to specify memory for squid
cache_mem 32 MB
cachedir ufs /var/spool/squid 1000 16 256

# allow assess to squid for all within our network
acl all src 0.0.0.0/0.0.0.0
http_access allow all
http_reply_access allow all

# internal host with no externally known name so we put
# our internal host name
visible_hostname hostname
# specifications needed for transparent proxy[2]
httpd_accel_port 80
httpd_accel_host virtual
httpd_accel_with_proxy on
httpd_accel_uses_host_header on
```

## Appendix E. /etc/raddb/clients.conf—Server #2

```
client 127.0.0.1 {
        secret    = password
        shortname       = localhost
        nastype = other
}
client 10.5.3.30 {
        secret = password
        shortname = other machine
}
```

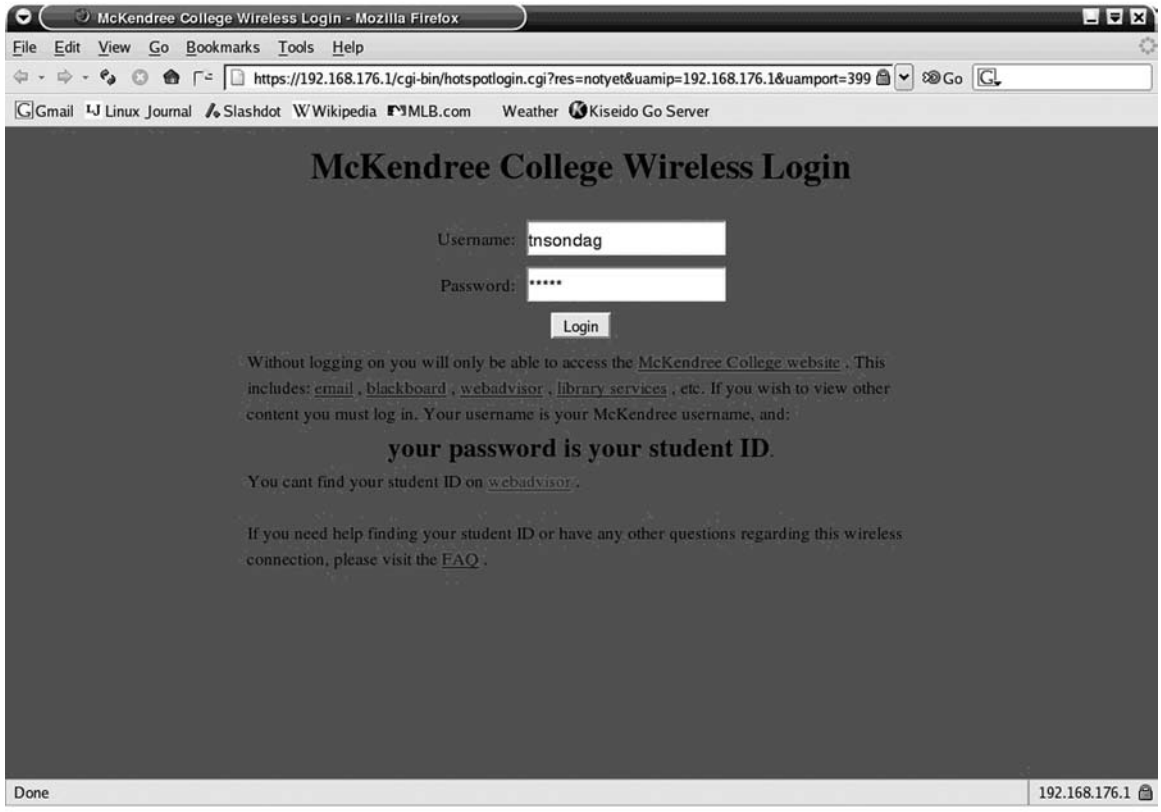
## Appendix F. /etc/raddb/users—Server #2

```
# example of an entry for a user
joeuser Auth-Type:=Local, User-Password==”passwd”
 Class = 0702345678,
 Session-Timeout = 3600,
 Idle-Timeout = 600,
 Acct-Interim-Interval = 60,
 WISPr-Bandwidth-Max-Up = 128000,
 WISPr-Bandwidth-Max-Down = 512000


# example of an entry for an access point
# The physical/mac address listed below is for the
# lan side of the router/access point
mac_address Auth-Type := Local, User-Password == “password”
 Framed-IP-Address = 192.168.182.10,
 Acct-Interim-Interval = 3600,
 Session-Timeout = 0,
 Idle-Timeout = 0
```

## Appendix G. /etc/chilli.conf—Server #3

```
# used to expand the network
net 192.168.176.0/20

# used to expand the number of hosts that can connect
# while still leaving a portion of the network for
# infrastructure
dynip 192.168.184.0/21

# used to give static addresses to the access points
statip 192.168.182.0/24

# internal DNS followed by external DNS
dns1 10.4.1.90
dns2 24.217.0.3

# radius server for the network
radiusserver1 10.4.1.90
radiusserver2 10.4.1.90

# radius secret used
radiussecret password

# interface Chillispot server to listens to DHCP requests
dhcpif eth1

# specified default login page
uamserver https://10.5.3.30/cgi-bin/hotspotlogin.cgi

# addresses that users can visit without authenticating
uamallowed 10.4.1.90,24.217.0.3,66.99.172.0/24

# this allows the access points to authenticate based on
# mac address only, this is required to log into the access
# points from the captive portal server
macauth

# this password corresponds with the password from the
# radius users file
macpasswd password
```

## Appendix H. Redirection page




## Appendix I. Method for flashing firmware of Linksys router

The firmware can be flashed using the built-in Web interface or via tftp. While help is available online[3] for this, the procedure outlined here may also be helpful. On newer versions of the Linksys routers, an older version of the Linksys firmware must be installed first that supports a bug in the ping function on the router. Once the older version is installed, you can exploit a bug in the ping command on the router to enable "boot wait," which enables the router to accept a connection to flash its firmware as it is booting.

Detailed instructions for this installation are as follows:

- First, download an old version of a Linksys firmware that supports the ping bug to enable boot wait. One is available at: ftp://ftp.linksys.com/pub/network/WRT54GS_3.37.2_US_code.zip
- Download and unzip this file.
- Plug an Ethernet patch cable into link #1 on the router (not the wan port) and the interface on your machine. Set the IP address of your computer to a static IP address in the 192.168.1.x range, not 192.168.1.1, which is used by the router.
- Log into router by opening a browser window and putting 192.168.1.1 into the address bar. (NOTE: This is only for factory preset routers.)
  Username: (*leave blank*)
  Password: admin
- Click on "administration".
- Click on "Firmware upgrade".
- Click "browse" and locate the old Linksys firmware on your machine.
- Click "upgrade".
- Wait patiently while it flashes the firmware….
- Click "setup".
- Click "basic setup".

- Choose "static ip" from the first box.
- For the IP address put in "10.0.0.1".
- For the netmask put in "255.0.0.0".
- For the gateway put in "10.0.0.2".
- You can leave everything else as their default settings.
- Choose save settings at the bottom of the page.
- Click on "administration".
- Click on "diagnostics".
- Click on "ping".

In the "address" box put the following commands in one at a time and click on "ping";

if you see the message that the host was unreachable you have done something wrong.

```
;cp${IFS}*/*/nvram${IFS}/tmp/n
;*/n${IFS}set${IFS}boot_wait=on
;*/n${IFS}commit
;*/n${IFS}show>tmp/ping.log
```

- After the last command you will see a list of all the nvram settings on the router, make sure that the line for "boot_wait" is set to on
- Unplug the router (the Linksys router will only look for new firmware on boot).
- Use tftp on your Linux or Windows machine.
- If the openwrt0-wrt54gs-squashfs.bin file is not in this directory, copy the file to this directory
- Run the following commands at the prompt (below are the Linux commands)

```
tftp 192.168.1.1
tftp> binary
tftp> rexmt 1
tftp> timeout 60
tftp> trace
tftp> put openwrt-xxx-x.x-xxx.bin
```

- The router will now reboot (it may take a very long time), when it is done rebooting, the DMZ light will turn off

The new firmware is now loaded onto the router.

## Appendix J. Nvram script for wireless routers

```
## server information stored as comments
##192.168.182.10 mainap 00:11:22:33:44:00
##192.168.182.11 cl202a 00:11:22:33:44:11
##192.168.182.20 lib01  00:11:22:33:44:22
##192.168.182.21 lib02  00:11:22:33:44:33
##192.168.182.22 lib03  00:11:22:33:44:44
##192.168.182.30 car01  00:11:22:33:44:55


## SAME for all
nvram set wl0_mode=ap
nvram set wl0_ssid=McK_Wireless
nvram set wl0_channel=9
nvram set lan_proto=dhcp

## Sample configuration for a few access points.
## Uncomment and run for the appropriate node.
## Make sure to
## add a line for every access point you have.

## UNIQUE for lib01
## allow connections to/from lib02, and lib03
#nvram set wl0_wds="00:11:22:33:44:33
00:11:22:33:44:44"

## UNIQUE for lib02
## allow connections to/from lib01
#nvram set wl0_wds="00:11:22:33:44:22"

## UNIQUE for lib03
## allow connections to/from lib01
#nvram set wl0_wds="00:11:22:33:44:22"

## SAME for all
nvram commit

## SAME for all
## This needed to be done to allow each wrt54gs router
## to accept an IP address from a DHCP server. This is
## only for the wrt54gs. Other access point/routers
## may require something different.
# cd /etc/init.d
# rm S05nvram
# cp /rom/etc/init.d/S05nvram .
# vi S05nvram
## place a # in front of (comment out)
## nvram set lan_proto="static"
```

## References


**1.** Thomas Eastep, "Port Knocking and Other Uses of 'Recent Match.'" www.shorewall.net/PortKnocking.html (accessed Aug. 11, 2006)

**2.** Ibid.

**3.** OpenWrtDocs/Installing-OpenWrt, wiki.openwrt.org/OpenWrtDocs/Installing (accessed Aug. 2, 2006).